UNIVERSIDADE FEDERAL
DE MINAS GERAIS

BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

LUIZ HENRIQUE GARIGLIO DOS SANTOS

PROJETO SISTEMA DE DESTRAVAMENTO PRESENCIAL SEQUENCIAL

MINAS GERAIS
2022

LUIZ HENRIQUE GARIGLIO DOS SANTOS

PROJETO SISTEMA DE DESTRAVAMENTO PRESENCIAL SEQUENCIAL

Descrição do desenvolvimento do projeto de um sistema de destravamento presencial sequencial para a disciplina de sistemas digitais.
Professor: Hermes Aguiar Magalhães

MINAS GERAIS
2022

Lista de ilustrações

Sumário

1 INTRODUÇÃO

Como forma de avaliação do conteúdo ministrado na disciplina de sistemas digitais pelo professor Hermes Aguiar Magalhães na Universidade Federal de Minas Gerais, foi proposto o desenvolvimento do proeto de um sistema de destravamento presencial sequencial.

Para desenvolvê-lo, foi utilizado o software Altera Quartus II, versão 13.0sp1. Desse modo, parte do trabalho foi realizada em VHDL e parte em esquemático.

A descrição do projeto consiste em projetar a lógica digital de um sistema que destranca um cofre quando estão presentes três pessoas e apertam 3 botões na ordem correta. Para atestar a presença dos três membros, é considerado que cada um possui um ”transpoder”, que emite um código formado pela primeira letra do nome do portador, codificada em ASCII, seguido do último dígito de seu número de matrícula, codificado em binário. A máquina não distingue letras maiúsculas e minúsculas.

Cada membro deve apertar seu botão na ordem correta e mantê-lo pressionado até que o cofre se abra. A indicação de que o próximo botão pode ser pressionado ocorre por um conjunto de três LEDs, que acendem sucessivamente a medida que as pessoas corretas apertam os botões corretos. Ao pressionar o último botão na ordem correta, a luz deve se manter acesa por no mínimo três segundos. A máquina de estados conta com uma entrada temporizada de onda quadrada com período de dois segundos.

2 PROJETO

Para o desenvolvimento do projeto, o primeiro passo foi determinar os três códigos para abertura do cofre, considerando meu nome e número de matrícula e os de mais dois colegas fictícios.

- Luiz - 202421137 - L7;
- Ana - 2022420292 - A2;
- Gabriel - 2022421523 - G3.

Passando as letras para ASCII, em binário, e os dígitos para binário:

- L: **0**1**0**0 1100;
- l: **0**11**1**0 1100;
- 7: 0111;
- A: **0**1**0**0 0001;
- a: **0**11**1**0 0001;
- 2: 0010;
- G: **0**1**0**0 0111;
- g: **0**11**1**0 0111;
- 3: 0011.

O primeiro dígito do código em ASCII é sempre zero, por isso pode ser omitido. Da mesma forma, o sexto dígito é o que difere as letras em maiúsculas e minúsculas. Como a máquina não deve fazer essa distinção, o dígito pode ser omitido, restando seis bits.

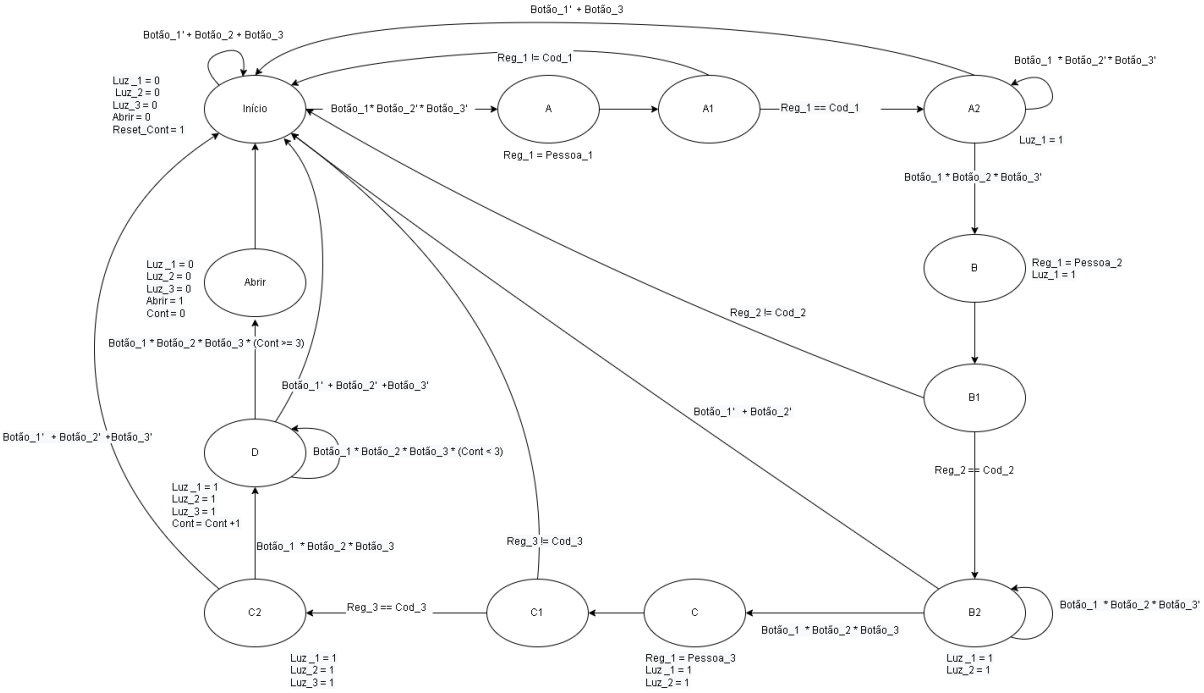
Para representar os dígitos de zero a nove em binário, são necessários quatro bits. Assim, os códigos são formados por dez bits e ficaram da seguinte forma:

- Luiz - 10 1100 0111 - botão 3;
- Ana - 10 0001 0010 - botão 1;
- Gabriel - 10 0111 0011 - botão 2.

2.1 DIAGRAMA DE ESTADOS

O próximo passo foi montar o diagrama de estados da máquina de estados, considerando todos os estados da máquina, as condições necessárias e as saídas de cada estado.

Figura 1 – Diagrama de Estados



Fonte: Autoria Própria, 2022

2.1.1 Início

É o estado inicial da máquina. Ele garante que as luzes são apagadas, o cofre fechado e o temporizador reiniciado. O estado não muda até que o primeiro botão seja pressionado.

2.1.2 Estados de Carregamento

São os estados nomeados como A, B e C. Ocorrem após um botão ser pressionado e neles, o código vindo do transpoder é carregado no registrador. Garante que o valor considerado na comparação é correto.

2.1.3 Estados de Teste

Ocorrem após os estados de carregamento, são A1, B1 e C1. O código armazenado no registrador é comparado com o código de referência e caso os códigos sejam iguais, a máquina vai para o próximo estado, caso contrário, retorna ao início.

2.1.4 Estados de Espera

A2, B2 e C2 ocorrem após os testes e só permitem mudança de estado caso um botão seja pressionado, se for correto, vai para o próximo, caso contrário, retorna ao início. Neles, os LEDs são acendidos.

2.1.5 Estágio D

É o último estado antes da abertura do cofre, ele garante que os LEDs se mantenham acesos por três segundos antes do cofre abrir.

2.1.6 Abrir

É o estado em que o cofre é efetivamente aberto e as luzes se apagam.

2.2 CAMINHO DE DADOS

Em seguida, comecei o projeto do caminho de dados que precisava de registradores, comparadores, mux e um contador(fornecido pelo professor). Os componentes foram então, desenvolvidos separadamente em VHDL.

2.2.1 Registrador

Usado para salvar o código que vem do transpoder com barramento de 9 bits na entrada e na saída. Também possui uma entrada de clock e uma de load.

Figura 2 – VHDL Registrador

```
library ieee;
use ieee.std_logic_1164.all;

entity Registrador is
    port
    (A: in std_logic_vector(9 downto 0);
    load, clk: in std_logic;
    s: out std_logic_vector(9 downto 0));
end Registrador;


architecture behavior of Registrador is
    begin
        process (clk, load)
            begin
                if (clk'event and clk = '1' and load = '1') then
                    s <= A;
                end if;
            end process;
    end behavior;
```

Fonte: Autoria Própria, 2022

2.2.2 Comparador

Possui duas entradas de dez bits e uma saída de um bit e compara o código do dispositivo com o que chega do transpoder.

Figura 3 – VHDL Comparador

```
library ieee;
use ieee.std_logic_1164.all;

entity Comparador is
    port
    (A,B: in std_logic_vector(9 downto 0);
    igual: out std_logic );
end Comparador;



architecture behavior of Comparador is

    begin
    igual <= '1' when A=B else '0';

end behavior;
```

Fonte: Autoria Própria, 2022

2.2.3 Comparador_Menor

Recebe o sinal do contador[1] e emite um sinal caso o sinal seja igual a três.

Figura 4 – VHDL Comparador_Menor

```
library ieee;
use ieee.std_logic_1164.all;

entity Comparador_Menor is
    port
    (A: in integer range 0 to 2;
    menor: out std_logic );
end Comparador_Menor;



architecture behavior of Comparador_Menor is

    begin
        menor <= '0' when A < 2 else '1';
    end behavior;
```

Fonte: Autoria Própria, 2022

[1] O projeto do contador foi fornecido pelo professor. Tendo sido necessária apenas uma pequena alteração para que o contador contasse até três para atender o requisito de tempo que os leds se mantivessem acesos.

2.2.4 Códigos

Armazena os códigos de referência para comparar com os recebidos pelo transpoder. Possui três entradas de endereço e uma saída de dez bits.

Figura 5 – VHDL Códigos

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Codigos is
    port
    (n1, n2, n3: in std_logic;
    s: out std_logic_vector (9 downto 0));
end Codigos;


architecture behavior of Codigos is
    type cod_pess is array(3 downto 0) of std_logic_vector(9 downto 0);
    signal cod : cod_pess := ("0000000000", "1000010010", "1001110011", "1011000111");
    signal endereco: std_logic_vector(1 downto 0);

    begin
       process (n1,n2,n3,cod, endereco)
       begin

          if (n3 = '0' and n2 = '0' and n1 = '1') then endereco <= "01";
          elsif (n3 = '0' and n2 = '1' and n1 = '0') then endereco <= "10";
          elsif (n3 = '1' and n2 = '0' and n1 = '0') then endereco <= "11";
          else endereco <= "00";
          end if;

          s <= cod(to_integer(unsigned(endereco)));
       end process;
    end behavior;
```

Fonte: Autoria Própria, 2022

2.2.5 MuxProjeto

Tem o funcionamento de um mux comum, com entradas e saída de dez bits.

Figura 6 – VHDL MuxProjeto

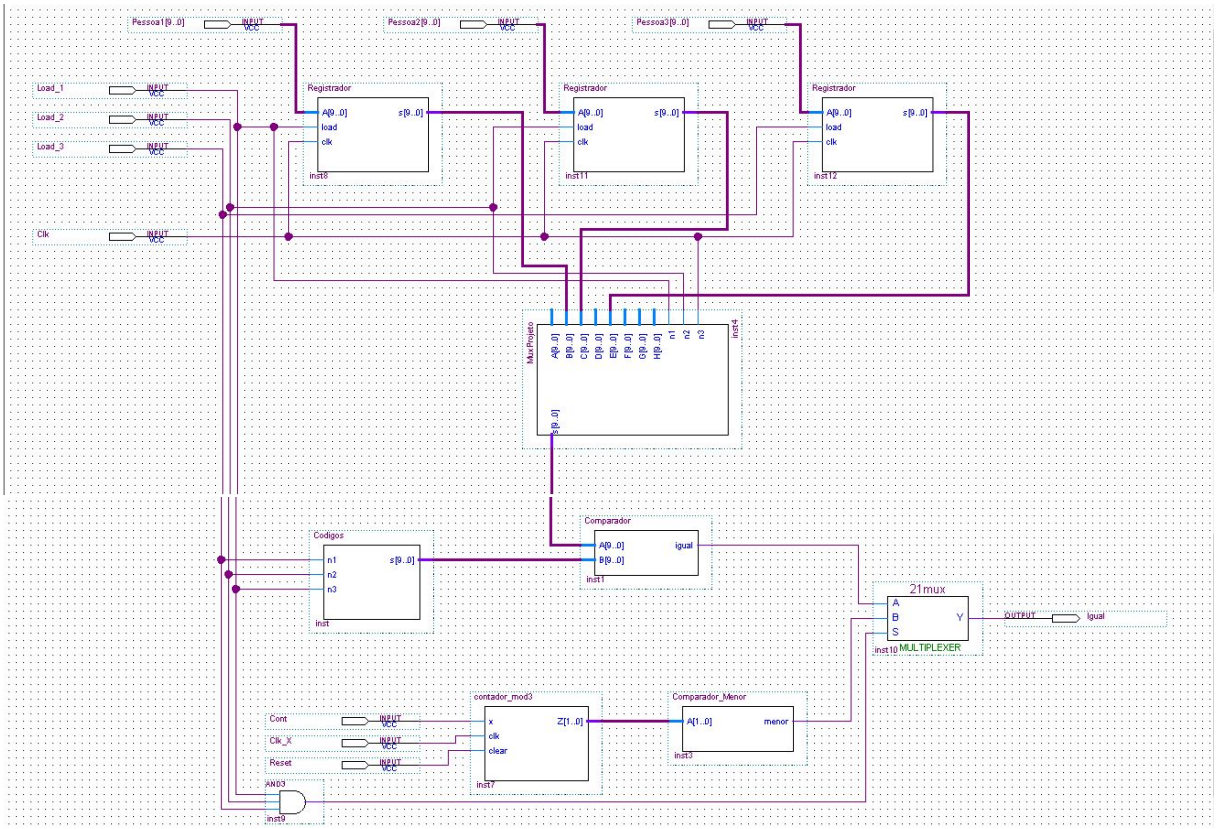
```
library ieee;
use ieee.std_logic_1164.all;

entity MuxProjeto is
    port
    (A,B,C,D,E,F,G,H: in std_logic_vector(9 downto 0);
    n1, n2, n3: in std_logic;
    s: out std_logic_vector (9 downto 0));
end MuxProjeto;



architecture behavior of MuxProjeto is

    begin
       process (n1,n2,n3,A,B,C,D,E,F,G,H)
       begin
          if (n3 = '0' and n2 = '0' and n1 = '0') then s <= A;
          elsif (n3 = '0' and n2 = '0' and n1 = '1') then s <= B;
          elsif (n3 = '0' and n2 = '1' and n1 = '0') then s <= C;
          elsif (n3 = '0' and n2 = '1' and n1 = '1') then s <= D;
          elsif (n3 = '1' and n2 = '0' and n1 = '0') then s <= E;
          elsif (n3 = '1' and n2 = '0' and n1 = '1') then s <= F;
          elsif (n3 = '1' and n2 = '1' and n1 = '0') then s <= G;
          elsif (n3 = '1' and n2 = '1' and n1 = '1') then s <= H;
          end if;
       end process;
    end behavior;
```

Fonte: Autoria Própria, 2022

2.2.6 CaminhoDados

Com os itens projetados, foi criado um símbolo para cada um deles. O caminho de dados então, foi montado na forma de esquemático realizando as conexões entre os componentes. Foram definidas entradas load1, load2, load3, clk, clk_x(entrada do temporizador), cont(inicia a contagem do tempo), reset(reseta a contagem de tempo), pessoa1, pessoa2 e pessoa3. Foi definida também uma única saída chamado igual.

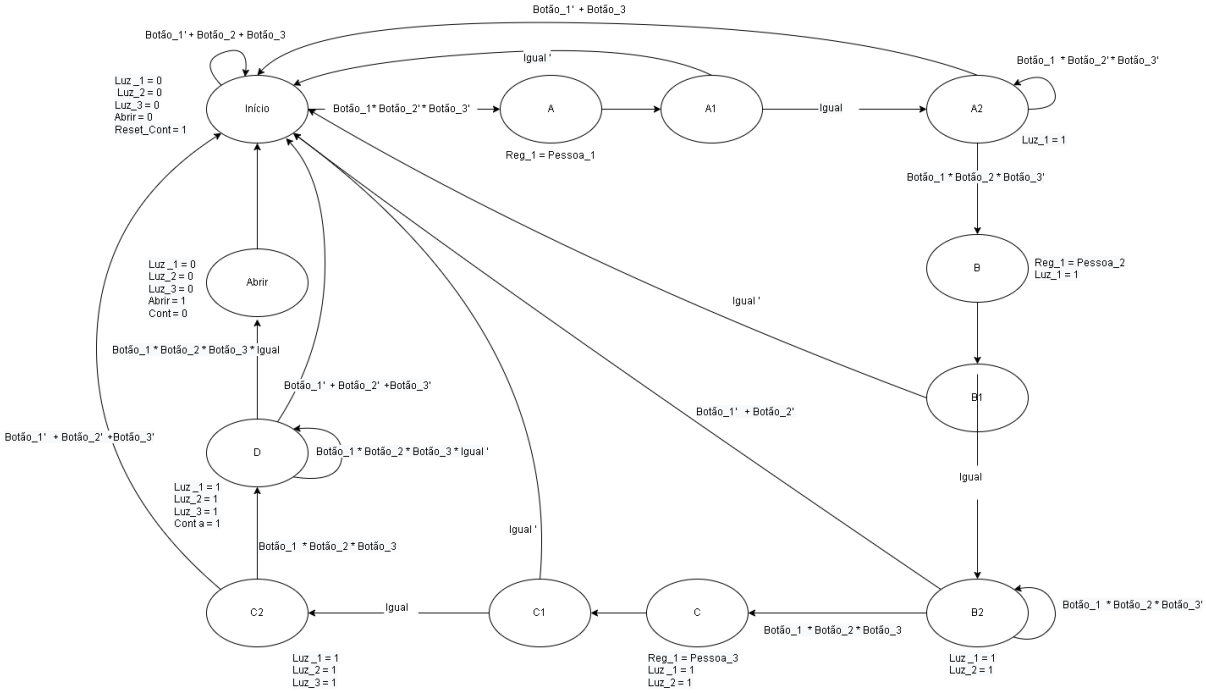
Figura 7 – Diagrama Esquemático CaminhoDados



Fonte: Autoria Própria, 2022

Com o projeto do caminho de dados pronto, foi possível refazer o diagrama de estados substituindo as funções complexas por sinais de alto nível.

Figura 8 – Diagrama de Estados Alto Nível



Fonte: Autoria Própria, 2022

2.3 CONTROLADORA

A controladora foi projetada em VHDL. Possui as entradas de clock, botão1, botão2, botão3 e igual e saídas luz1, luz2, luz3, load1, load2, load3, conta, reset_tempo e abre.

Para definir o comportamento, foi definido um tipo estado com todos osestados possíveis para a máquina e determinados dois *"process"*, um sensível a borda de clock e outro que determina as mudanças de estado e as saídas de cada estado. Para ficar mais fácil o controle de quais saídas de cada estado e as condições de mudança de estado, montei a seguinte tabela.

Figura 9 – Tabela de Estados

Estado_Atual	B1	B2	B3	Igual	Proximo_Estado	Luz1	Luz2	Luz3	Load1	Load2	Load3	Abre	Conta	Reset_Tempo
Início	0	X	X	X	Início	0	0	0	0	0	0	0	0	1
Início	1	0	0	X	A	0	0	0	0	0	0	0	0	1
Início	X	X	1	X	Início	0	0	0	0	0	0	0	0	1
Início	X	1	X	X	Início	0	0	0	0	0	0	0	0	1
A	X	X	X	X	A1	0	0	0	1	0	0	0	0	0
A1	X	X	X	0	Início	0	0	0	0	0	0	0	0	0
A1	X	X	X	1	A2	0	0	0	0	0	0	0	0	0
A2	0	X	X	X	Início	1	0	0	0	0	0	0	0	0
A2	1	0	0	X	A2	1	0	0	0	0	0	0	0	0
A2	X	X	1	X	Início	1	0	0	0	0	0	0	0	0
A2	1	1	0	X	B	1	0	0	0	0	0	0	0	0
B	X	X	X	X	B1	1	0	0	0	1	0	0	0	0
B1	X	X	X	0	Início	1	0	0	0	0	0	0	0	0
B1	X	X	X	1	B2	1	0	0	0	0	0	0	0	0
B2	0	X	X	X	Início	1	1	0	0	0	0	0	0	0
B2	X	0	X	X	Início	1	1	0	0	0	0	0	0	0
B2	1	1	0	X	B2	1	1	0	0	0	0	0	0	0
B2	1	1	1	X	C	1	1	0	0	0	0	0	0	0
C	X	X	X	X	C1	1	1	0	0	0	1	0	0	0
C1	X	X	X	0	Início	1	1	0	0	0	0	0	0	0
C1	X	X	X	1	C2	1	1	0	0	0	0	0	0	0
C2	0	X	X	X	Início	1	1	1	0	0	0	0	0	0
C2	X	0	X	X	Início	1	1	1	0	0	0	0	0	0
C2	X	X	0	X	Início	1	1	1	0	0	0	0	0	0
C2	1	1	1	X	D	1	1	1	0	0	0	0	0	0
D	0	X	X	X	Início	1	1	1	1	1	1	0	1	0
D	X	0	X	X	Início	1	1	1	1	1	1	0	1	0
D	X	X	0	X	Início	1	1	1	1	1	1	0	1	0
D	1	1	1	0	D	1	1	1	1	1	1	0	1	0
D	1	1	1	1	Abrir	1	1	1	1	1	1	0	1	0
Abrir	X	X	X	X	Início	0	0	0	0	0	0	1	0	0

Fonte: Autoria Própria, 2022

2.4 PROJETO COFRE

Por fim, para conectar a controladora ao caminho de dados utilizei o VHDL. As entradas definidas foram clock, temporizador X, botões (B1, B2 e B3), e Pessoa1, Pessoa2 e Pessoa3 (de dez bits). As saídas foram abrir, led1, led2 e led3.

Na arquitetura foi necessário declarar a controladora e o caminho de dados como componentes e então, realizar as conexões entre eles.

Figura 10 – VHDL Projeto_Cofre

```
library ieee;
use ieee.std_logic_1164.all;

entity Projeto_Cofre is
    port
    (Clock, X: in std_logic;
    B1,B2,B3: in std_logic;
    Pessoa1, Pessoa2, Pessoa3: in std_logic_vector(9 downto 0);
    Abrir, luz_1, luz_2, luz_3: out std_logic
    );
end Projeto_Cofre;
architecture behavior of Projeto_Cofre is

    component Controladora is
       port(clk: in std_logic;
          botao1,botao2,botao3, Igual: in std_logic;
          Luz1, Luz2, Luz3, Load1, Load2, Load3, Conta, Reset_Tempo, Abre: out std_logic
       );
    end component;

    component CaminhoDados is
       port(Clk, Clk_X: in std_logic;
          Load_1, Load_2, Load_3, Cont, Reset: in std_logic;
          Igual: out std_logic
          );
    end component;

    signal F_igual, F_load1, F_load2, F_load3, F_conta, F_reset: std_logic;

    begin
       control: Controladora port map(
       botao1 => B1,botao2 => B2,botao3 => B3, clk => Clock,
       Luz1 => luz_1,Luz2 => luz_2,Luz3 => luz_3,Abre => Abrir,
       Igual => F_igual,
       Load1 => F_load1,Load2 => F_load2,Load3 => F_load3,
       Conta => F_conta,Reset_Tempo => F_reset);

       cam_dados: CaminhoDados port map(
       Clk => Clock,Clk_X => X,
       Load_1 => F_load1,Load_2 => F_load2,Load_3 => F_load3, Cont => F_conta, Reset => F_reset,
       Igual => F_igual);
end behavior;
```

Fonte: Autoria Própria, 2022

Referências

[1] PEDRONI, V.A; Eletrônica Digital Moderna e VHDL. RJ, 2010. Acesso em: 09 de set. de 2022;

[2] CÔRTES, Mario. VHDL: Circuitos Combinacionais. Unicamp, 2011. Disponível em ≪https://www.ic.unicamp.br/ cortes/mc602/slides/VHDL/VHDL_2_MC_Circ_Comb_v2.pdf≫. Acesso em: Acesso em: 09 de set. de 2022.